In the Specification

Please amend the specification of this application as follows:

Rewrite the paragraph at page 1, lines 5 to 11 as follows:

--This application is a divisional of copending U.S. Serial No. 09/798,561 ~~(Docket No. TI 30485)~~ filed on March 2, 2001 and incorporated herein by reference. U.S. Serial No. 09/798,561 claims the priority under 35 U.S.C. 119(e)(1) of the following co-pending U.S. provisional applications: 60/186,326 ~~(Docket TI 30526)~~ filed on March 2, 2000; and 60/219,340 ~~(Docket TI 30498)~~ originally filed on March 2, 2000 as non-provisional U.S. Serial No. 09/515,093 and thereafter converted to provisional application status by a petition granted on August 18, 2000.--

Rewrite the paragraph at page 2, lines 12 to 17 as follows:

--Functional testing, wherein a designer is responsible for generating test vectors that are intended to ensure conformance to specification, still remains a widely used test methodology. For very large systems this method proves inadequate in providing a high level of detectable fault coverage. Automatically generated test ~~patters~~ patterns would be desirable for full testability, and controllability and observability are key goals that span the full hierarchy of test (from the system level to the transistor level).--

Rewrite the paragraph at page 2, line 18 to page 3, line 3 as follows:

--Another problem in large designs is the long time and substantial expense involved. It would be desirable to have testability circuitry, system and methods that are consistent with a concept of design-for-reusability. In this way, subsequent devices and systems can have a low marginal design cost for

testability, simulation and emulation by reusing the testability, simulation and emulation circuitry, systems and methods that are implemented in an initial device. Without a proactive testability, simulation and emulation approach, a large amount of subsequent design time is expended on test pattern creation and upgrading.--

Rewrite the paragraph at page 5, lines 11 to 20 as follows:

--To combat this trend, system designers have worked to keep these buses exposed, building system components in a way that enabled the construction of prototyping systems with exposed buses. This approach is also under siege from the ever-increasing march of system clock rates. As CPU clock rates increase, chip to chip interface speeds are not keeping pace. Developers find that a partitioned system's performance does not keep pace with its integrated counterpart, due to interface wait states added to compensate for lagging chip to chip communication rates. At some point, this performance degradation reaches intolerable levels and the partitioned prototype system is no longer a viable debug option. We have entered an era where production devices must serve as the platform for application development.--

Rewrite the paragraph at page 11, line 14 as follows:

--2. Real-time Data Exchange (~~RTDX~~ RTDX™ a trademark of Texas Instruments Incorporated);--

Rewrite Table 1 at page 12 as follows:

--Table 1. **Emulation System Architecture and Usage**

Architectural Component	Visibility Provisions	Control Provisions	Debug Usage
RTE	Static view of the CPU and memory state after background program is stopped. Interrupt driven code continues to execute.	Analysis components are used to stop execution of background program.	Basic debug Computational problems Code design problems
~~RTDX~~ RTDX™	Debugger software interacts with the application code to exchange commands and data while the application continues to execute.	Analysis components are used to identify observation points and interrupt program flow to collect data.	Dynamic instrumentation Dynamic variable adjustments Dynamic data collection
Trace	Bus snooper hardware collects selective program flow and data transactions for export without interacting with the application.	Analysis components are used to define program segments and bus transactions that are to be recorded for export.	Prog. Flow corruption debug Memory corruption Benchmarking Code Coverage Path Coverage Program timing problems
Analysis	Allows observation of occurrences of events or event sequences. Measure elapsed time between events. Generate external triggers.	Alter program flow after the detection of events or event sequences.	Benchmarking Event/sequence identification Ext. trigger generation Stop program execution Activate Trace and ~~RTDX~~ RTDX™

--

Rewrite the paragraph at page 13, lines 3 to 10 as follows:

--RTDX™ provides real-time data transfers between an emulator host and target application. This component offers both bi-directional and uni-directional DSP target/host data transfers facilitated by the emulator. The DSP (or target) application may collect target data to be transferred to the host or receive data from the host, while emulation hardware (within the DSP and the emulator) manages the actual transfer. Several ~~RTDX~~ RTDX™ transfer mechanisms are supported, each providing different levels of bandwidth and pin utilization allowing the trade off of gates and pin availability against bandwidth requirements.--

Rewrite the paragraph at page 14, lines 19 to 21 as follows:

--The ~~RTDX~~ RTDX™ and Trace functions provide similar, but different forms of visibility. They differ in terms of how data is collected, and the circumstances under which they would be most effective. A brief explanation is included below for ~~clarity:~~ clarity.-

Rewrite the paragraph at page 14, line 22 to page 15, line 3 as follows:

--RTDX™ (Real Time Data eXchange) is a CPU assisted solution for exchanging information; the data to be exchanged have a well-defined behavior in relation to the program flow. For example, ~~RTDX~~ RTDX™ can be used to record the input or output buffers from a DSP algorithm. ~~RTDX~~ RTDX™ requires CPU assistance in collecting data hence there is definite, but small, CPU bandwidth required to accomplish this. Thus, ~~RTDX~~ RTDX™ is an application intrusive mechanism of providing visibility with low recurring overhead cost.--

Rewrite the paragraph at page 15, lines 11 to 14 as follows:

--Trace data is unidirectional, going from target to host only. ~~RTDX~~ RTDX™ can exchange data in either direction although unidirectional forms of ~~RTDX~~ RTDX™ are supported (data logging). The Trace data path can also be used to provide very high speed uni-directional ~~RTDX~~ RTDX™ (CPU collected trace data).--

Rewrite the paragraph at page 15, line 15 as follows:

--The high level features of Trace and ~~RTDX~~ RTDX™ are outlined in Table 2.--

Rewrite Table 2 at page 15 as follows:

--Table 2. ~~RTDX~~ RTDX™ and Trace Features

Features	RTDX™	Trace
Bandwidth/pin	Low	High
Intrusiveness	Intrusive	Non-intrusive
Data Exchange	Bi-directional or uni-directional	Export only
Data collection	CPU assisted	CPU or Hardware assisted
Data transfer	No extra hardware for minimum BW (optional hardware for higher BW)	Hardware assisted
Cost	Relatively low recurring cost	Relatively high recurring cost

--

Rewrite the paragraph at page 16, lines 1 to 9 as follows:

--Advanced analysis provides a non-intrusive on-chip event detection and trigger generation mechanism. The trigger outputs created by advanced analysis control other infrastructure components such as Trace and ~~RTDX~~ RTDX™. Historical trace technology used bus activity exported to a logic analyzer to generate triggers that controlled trace within the logic analyzer unit or generated triggers which were supplied to the device to

halt execution. This usually involved a chip that had more pins than the production device (an SE or special emulation device). This analysis model does not work well in the System-on-a-Chip (SOC) era as the integration levels and clock rates of today's devices preclude full visibility bus export.--

Rewrite the paragraph at page 15, lines 10 to 19 as follows:

--Advanced analysis provides affordable on-chip instruction and data bus comparators, sequencers and state machines, and event counters to recreate the most important portions of the triggering function historically found off chip. Advanced analysis provides the control aspect of debug triggering mechanism for Trace, ~~RTDX~~ RTDX™ and Real-Time Emulation. This architectural component identifies events, tracks event sequences, and assigns actions based on their occurrence (break execution, enable/disable trace, count, enable/disable ~~RTDX~~ RTDX™, etc.). The modular building blocks for this capability include bus comparators, external event generators, state machines or state sequencers, and trigger generators. The modularity of the advanced analysis system allows the trade off of functionality versus gates.--

Rewrite the paragraph at page 18, line 5 as follows:

--• ~~RTDX~~ RTDX™;--

Rewrite the paragraph at page 19, lines 12 to 17 as follows:

--The emulation controller 12 accesses Real-time Emulation capabilities (execution control, memory, and register access) via a 3, 4, or 5 bit scan based interface. ~~RTDX~~ RTDX™ capabilities can be accessed by scan or by using three higher bandwidth ~~RTDX~~ RTDX™ formats that use direct target-to-emulator connections other than scan. The input and output triggers allow other system components to signal the chip with debug events and vice-versa.--

Rewrite the paragraph at page 20, lines 4 to 8 as follows:

--The emulation controller 12 communicates with the target system 16 through a target cable or cables at 17. Debug, Trace, Triggers, and ~~RTDX~~ RTDX™ capabilities share the target cable, and in some cases, the same device pins. More than one target cable may be required when the target system deploys a trace width that cannot be accommodated in a single cable. All trace, ~~RTDX~~ RTDX™, and debug communication occurs over this link.--

Rewrite the paragraph at page 29, line 15 to page 30, line 4 as follows:

--FIGURE 11, when taken in conjunction with FIGURE 8, illustrates pertinent portions of further exemplary embodiments of the trace collector 21 of FIGURE 2. The embodiment of FIGURE 11 includes a memory access trace packet generator 111 which can produce a data/address trace packet stream (such as illustrated in FIGURE 9) and a memory reference sync point (such as illustrated in FIGURE 10). The memory access trace packet generator 111 of FIGURE 11 is coupled for input from the PC register, and also receives data/address information ~~112~~ from the target processor core. The memory access trace packet generator 111 also receives trigger information, for example conventionally generated trigger information, which designates when to begin and end memory access trace activity. The memory access trace packet generator 111 is also coupled to the table of ID numbers at 83, so the memory reference sync point of FIGURE 10 can be provided with the proper PC sync ID number.--

Rewrite the paragraph at page 30, lines 5 to 8 as follows:

--In response to the trigger information, the memory access trace packet generator 111 can produce from the data/address

information ~~112~~ a data/address trace packet stream. This packet stream is provided to the stream combiner 85 of FIGURE 8, for inclusion in the composite packet stream of FIGURE 8.--

Rewrite the paragraph at page 30, line 9 to page 31, line 2 as follows:

--The packet generator 111 also receives at 115 information (e.g., from the PC trace packet generator 82 of FIGURE 8) indicative of the issuance of a PC sync packet. In response to this information at 115, the memory access trace packet generator 111 retrieves the current PC sync ID number from the table 83, and produces (as needed) a memory reference sync point such as illustrated in FIGURE 10. The occurrence of a PC sync point also clears a counter 112 that is incremented each time the PC register is loaded. Thus, the counter 112 provides a running record of the number of new PC loads since the last PC sync point. Thus, the count output of the counter 112 indicates a number of PC loads from which the current PC value is offset from the last PC sync point. Thus, when PC trace is active, indicated by a signal ~~118~~ (for example from PC trace packet generator 82 of FIGURE 8), the memory access trace packet generator 111 can, within a command such as illustrated in FIGURE 9, identify the corresponding PC by this offset value rather than by the entire native PC value, which advantageously reduces the amount of information in (and hence the bandwidth required by) the memory reference command of FIGURE 9. The native PC value can be included in the FIGURE 9 command if PC trace is inactive.--

Rewrite the paragraph at page 40, lines 10 to 15 as follows:

--FIGURES 25-27 are similar to FIGURE 24 and illustrate exemplary operations which can be performed by the transmission formatter of FIGURES 22 and 22A when additional trace packet data

is required but none is available ~~form~~ from the FIFO. In FIGURE 25, the transmission formatter simply stalls until enough additional trace packet data (from the next trace packet) is available (at 251) to build a complete 6-bit transmission packet.--